TARTU ÜLIKOOL
Arvutiteaduse instituut
Informaatika õppekava

Age Roosi

Thonny logifailide analüüsi automatiseerimine

Bakalaureusetöö (9 EAP)

Juhendajad: Eno Tõnisson, PhD
Heidi Meier, MSc

Tartu 2019

Thonny logifailide analüüsi automatiseerimine

Lühikokkuvõte:

Programmeerimisülesannete lahendamise protsessi uurimine võib anda kasulikku informatsiooni õppijate käitumismustrite kohta ja õppematerjalide täiustamiseks. Thonny logifailide analüüsimine on üks võimalus vajalike andmete kogumiseks. Kuna logifailid on mahukad, siis on mõtet analüüs automatiseerida.

Siinse bakalaureusetöö raames koostati tarkvara Thonny logifailide automaatseks analüüsimiseks, mille abil tuuakse välja näiteks õppijal esinenud veateated, kleebitud tekstid ja ülesannete lahendamise aeg. Tarkvaral on kaks väljundit – graafiline liides ja CSV-failid.

Võtmesõnad:

Thonny, logifail, andmetöötlus, programmeerimine

CERCS: P175 Informaatika, süsteemiteooria, S281 Arvuti õpiprogrammide kasutamise metoodika ja pedagoogika

Automating Thonny Log File Analysation

Abstract:

Investigating the process of solving programming tasks may provide useful information about study materials and learners' behaviour patterns. One opportunity how to collect needed data is to analyse Thonny log files. The analysation should be automated because log files mostly contain a large amount of data.

Data analysis software for automatically analysing Thonny log files was created in this thesis. The software enables to get analysation that shows for example learners' error messages, pasted texts and the time spent on solving the tasks. Software has two outputs – graphical user interface and CSV files.

Keywords:

Thonny, log file, data analysation, programming

CERCS: P175 Informatics, systems theory, S281 Computer-assisted education

Sisukord

1. Sissejuhatus 4

2. Programmeerimiskeskkond Thonny 6

3. Ülevaade Thonny logifailidest 9

3.1 Thonny logifailid 9

3.2 Thonny logifailide struktuur 9

3.2.1 Teksti sisestamine ja kustutamine 10

3.2.2 Veateated 13

3.2.3 Käivitamistega seotud kirjed 14

4. Thonny logifailide analüsaator 16

4.1 Tarkvara funktsionaalsused ja kasutamine 16

4.1.1 Analüüs graafilises liideses 17

4.1.1.1 Alamleht „Üldinfo“ 17

4.1.1.2 Alamleht „Veatüübid“ 18

4.1.1.3 Alamleht „Käivitamised“ 19

4.1.1.4 Alamlehed tabelitega 20

4.1.2 CSV-failid 21

4.2 Tagasiside tarkvarale 22

4.3 Edasised võimalused 24

5. Kokkuvõte 26

Viidatud kirjandus 27

Lisad 28

I. Thonny logifailide analüsaator 28

II. Litsents 29

1. **Sissejuhatus**

Tartu Ülikoolis on mitmeid programmeerimise algkursuseid. Programmeerimise algkursused on näiteks “Programmeerimine”, mis sisaldub üheksas õppekavas, ning “Programmeerimise alused”, mis on 22 õppekavas [1]. Samuti on Tartu Ülikoolis mitu programmeerimise vaba juurdepääsuga kursust (MOOC) algajatele [2]. Algajatele mõeldud programmeerimise kursustel osalejate arv on viimastel aastatel järsult tõusnud. Algkursustel osaleb sadu õppijaid ja programmeerimise vaba juurdepääsuga kursustel tuhandeid. Tartu Ülikooli statistika kohaselt [3] on informaatika bakalaureuseõppe üliõpilaste arv kasvanud rohkem kui kolm korda võrreldes aastaga 2011.

Õpetaja või õppejõud ei jõua korraga kõiki õppijaid jälgida, kui neid on palju. Programmi loomise protsess on igal lahendajal erinev ning seetõttu on keerukas nende raskustele ja probleemidele õigeid lahendusi välja pakkuda. Programmeerimise puhul võib juba väike eksimus tekitada programmi töös vigu ja kui eksimused on korduvad, siis võivad mitmed kodutööd ja kontrolltööd jääda selle tõttu poolikuks. Masinlikult lahendamise protsessi analüüsides on võimalik avastada õppija probleeme. Samuti saab uurides paljude õppijate lahenduskäike parandada materjale, ülesandeid ja õppemetoodikat.

Programmeerimise õppimist on võimalik alustada eri keeltes, kuid Python on muutunud põhiliseks keeleks, mida kasutatakse programmeerimise sissejuhatavatel kursustel [4]. Selles keeles toimub algõpe ka Tartu Ülikoolis. Tartu Ülikoolis kasutavad algajad programmeerimiskeskkonda Thonny, mille põhiautor on Tartu Ülikooli endine õppejõud Aivar Annamaa [5]. Thonny loob iga kasutussessiooni kohta logi, kuhu koos ajatemplitega salvestatakse näiteks kasutaja iga klahvivajutus, liikumine eri akende vahel ning esinevad veateated.

Eelnimetatud logifailid annavad hea võimaluse analüüsida õppija programmiloomise protsessi ning leida sealt häid ning halbu külgi. Logifaile ja nendes kasutaja tegevuse kohta olevaid kirjeid on analüüsinud Keili Pedel oma bakalaureusetöös “E-kursuse "Programmeerimise alused" logifailide analüüs” [6]. Täpsemalt õppijate käitumismustreid on logifailide abil analüüsitud Heidi Meieri magistritöös “Õppijate käitumismustrid programmeerimisülesande lahendamisel: logifailide analüüs”, kus ta leidis, et õppijatel, kes ei töötanud järk-järgult, läks kõige rohkem aega ja oli palju ning korduvaid veateateid [7]. Siiani on logifailide analüüsimine olnud pigem ajamahukas protsess. Selleks, et analüüs oleks mugavam igale õppejõule ja seda

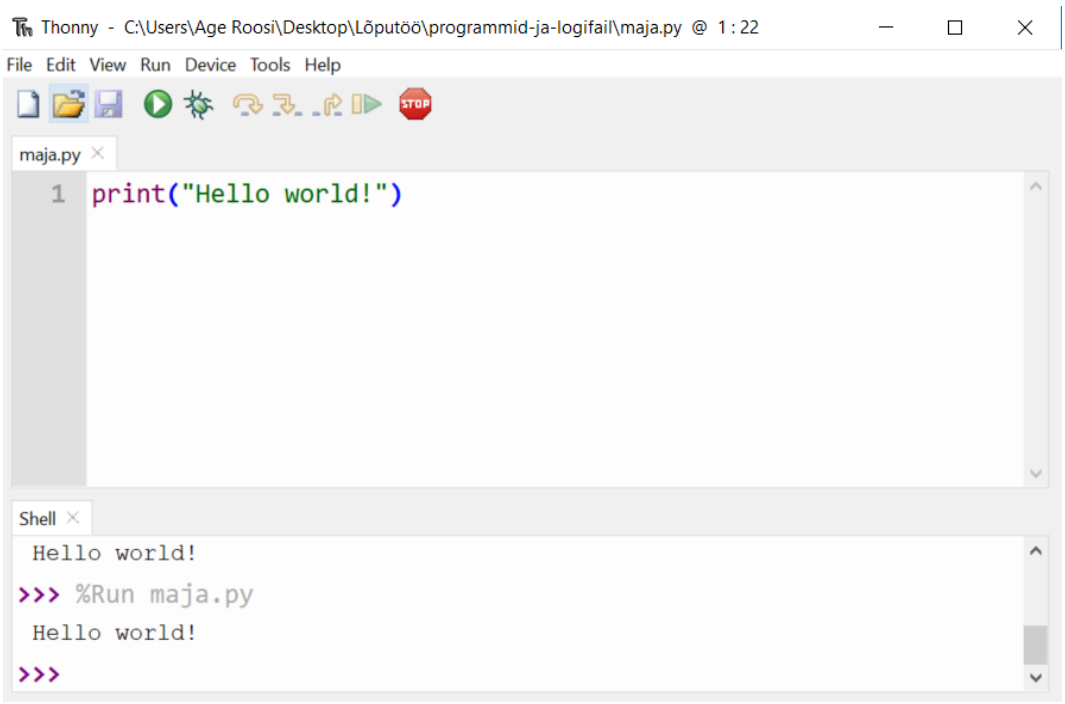
võimalust tihedamini rakendataks, oleks kasulik Thonny logifailide analüüsimist püüda automatiseerida.

Selle bakalaureusetöö eesmärk on luua tarkvaralahendus, millega saaks kasutaja logifailides leiduvate andmete põhjal automaatselt koostada esmase analüüsi. Selle abil oleks tal võimalik teatud aspektides hinnata ülesande lahendamise protsessi ilma, et peaks ise logifaile eraldi avama ja nende sisu uurima. Tarkvaralahenduse kasutajateks on õppejõud ja õpetajad, kes õpetavad programmeerimist programmeerimiskeskkonnas Thonny, ning uurijad, kes soovivad näiteks analüüsida õppijate programmeerimise protsessi või õppematerjale, mille põhjal lahendati ülesandeid Thonnys.

Bakalaureusetöö esimeses osas antakse ülevaade Thonny ning Thonny logifailide kohta. Töö teises osas kirjeldatakse loodud tarkvaralahendust. Lõpuks analüüsitakse õppejõududelt tarkvaralahendusele saadud tagasisidet ja pakutakse välja ideid edasiseks arendamiseks.

2. Programmeerimiskeskkond Thonny

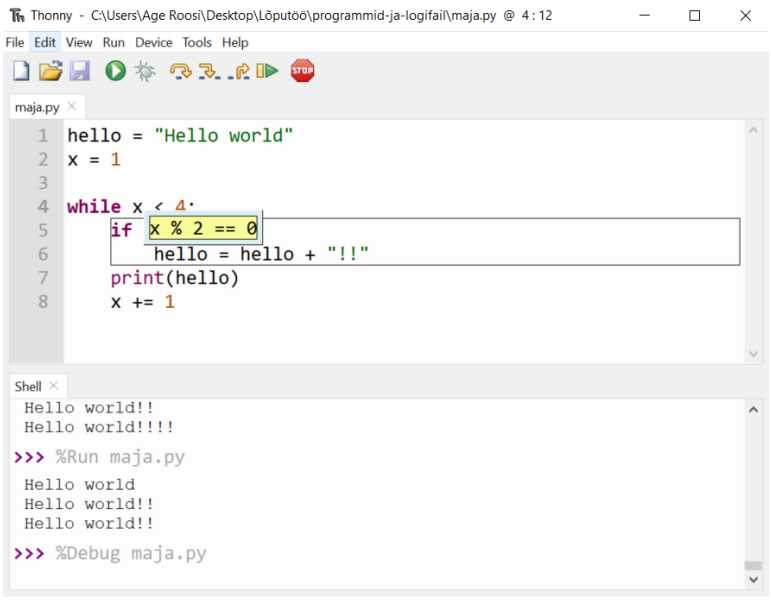
Thonny[1] on algajatele mõeldud programmeerimiskeskkond, mille abil on lihtne alustada programmeerimise õppimist. Selles on üks uuemaid Pythoni versioone (3.7) sisseehitatud ning kasutaja ei pea oma arvutisse eraldi Pythonit paigaldama. Kasutajaliidesesse (joonis 1) ei ole lisatud funktsionaalsusi, mis võivad algajatele keeruliseks osutuda ning seetõttu tähelepanu programmeerimiskeele õppimiselt mujale juhtida.



Joonis 1. Thonny kasutajaliides.

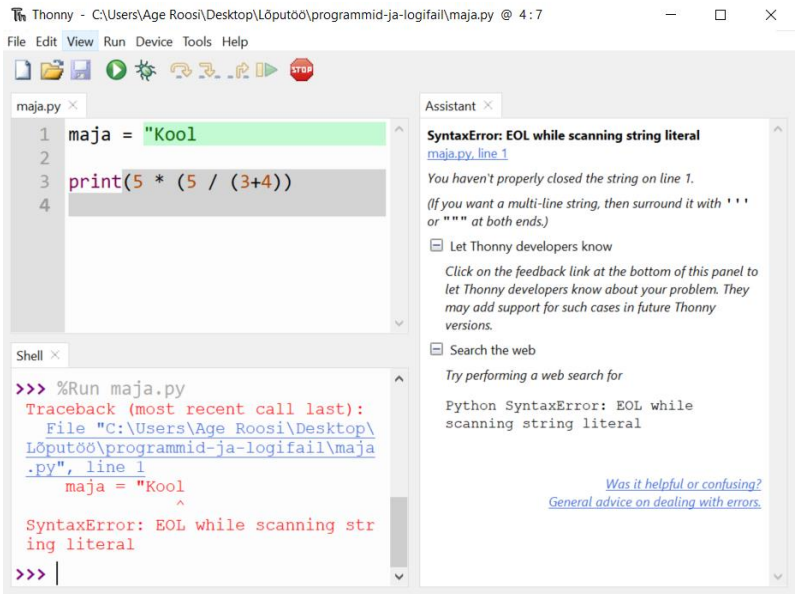
Samuti on arenduskeskkonnal lihtne silur (moodul tarkvara silumiseks, ingl *debugger*), millega on võimalik vaadata programmi tööd samm-sammult ilma katkestuspunkte (ingl *breakpoint*) lisamata (joonis 2). Silur läbib programmikoodi struktuuri, mitte lihtsalt järjest koodi ridu. Siluri kasutamise võimalus on oluline õppijale, et saada paremini aru koodi loogikast ning mõista enda programmi töös esinenud vigu.

[1] Vt https://thonny.org/.



Joonis 2. Silumine Thonnys.

Olulist tähelepanu pööratakse vigadele ning õppija abistamisele probleemide lahendamisel. Kasutajale tuuakse kasutajaliideses esile süntaksivead nagu sulgemata jutumärgid, ülakomad ja sulud, et viga oleks võimalik märgata juba enne programmi käivitamist. Käsureal kuvatakse programmi käivitamisel tekkinud vea kohta teade ning tuuakse esile ka vea põhjustanud programmilõik ning täpne rida (joonis 3). Üks uuemaid funktsionaalsusi on pärast käivitamist avanev aken "Assistant", kus kuvatakse veelgi soovitusi probleemi lahendamiseks ning pakutakse välja ühelauseline juhis, mille abil otsida internetist lisainformatsiooni.



Joonis 3. Programmeerija abistamine kasutajaliideses.

Thonny põhiline arendus toimus Tartu Ülikooli arvutiteaduse instituudis ning selles programmeerimiskeskkonnas õpivad programmeerimiskursustel osalejad. Selleks, et programmeerimiskeskkond oleks kasutajale veelgi mugavam, täiustavad programmeerimiskeskkonda Aivar Annamaa ja ka teised tema juhendamisel.

3. Ülevaade Thonny logifailidest

Järgnevas peatükis antakse ülevaade Thonny logifailide struktuurist ja olemusest. Samuti kirjeldatakse analüüsiks kasutatavaid logifaili kirjeid täpsemalt.

3.1 Thonny logifailid

Programmeerimiskeskkonnal Thonny on olemas põhjalik logifailide süsteem, mille abil salvestatakse kasutaja kõik tegevused ja nendele järgnevad sündmused nagu näiteks veateated. Logimist alustatakse programmeerimiskeskkond Thonny avamisel ning selle sulgemisel luuakse uus logifail. Seega on ühes logifailis andmed ühe kasutussessiooni kohta. Logifail salvestatakse txt-failivormingus kohalikule kõvakettale programmi Thonny kaustas asuvasse alamkausta „user_logs“ ning sellesse faili salvestatakse programmi kasutamise kohta kogu informatsioon koos ajatemplitega. Thonny logifailides kasutatakse lihtsustatud andmevahetus-vormingut JSON (ingl *Javascript Object Notation*) [8].

Ühes logifailis võib olla mitme programmeerimisülesande lahendusi, kui lahendatakse mitut ülesannet ning ülesannete vahetamisel ei panda programmeerimiskeskkonda kinni. Samas võivad ühe ülesande logid olla jagunenud mitme logifaili vahel, kui lahendaja paneb programmeerimiskeskkonna vahepeal kinni ning siis avab uuesti. Parem on analüüsida logifaile, mis sisaldavad terve ülesande lahendust, kuna saame kindlad olla, et kogu informatsioon ülesande lahendamise kohta on logifailis olemas ja seetõttu ei lähe osa informatsiooni kaduma.

Kontrolltööde puhul on lahendajatel kindel algus- ja lõppaeg. Kontrolltöö logifailidest on ajatemplite abil võimalik välja sorteerida ainult kontrolltöö kohta käivad logid ning lahendatavate ülesannete kohta on kogu informatsioon olemas. Samuti on õppijatel samasugused tingimused kontrolltöö lahendamiseks: kindel ajavahemik, sarnased ruumid ja iseseisev lahendamine. Kontrolltööde logifailid on seega analüüsimiseks kõige lihtsamad, kuna andmestik on ühtlane ja õppijad lahendavad ülesandeid samades tingimustes.

3.2 Thonny logifailide struktuur

Väga põhjalikult on logifaile analüüsinud Heidi Meier oma magistritöös, kus ta keskendub õppijate käitumismustrite kindlaks tegemisele [7]. Keili Pedeli bakalaureusetöös analüüsiti e-kursuse „Programmeerimise alused“ logifaile. Ta jagab logifailides olevad kirjed viide gruppi:

Thonny programmiaknaga, programmikoodiga, failidega seotud kirjed, programmi käivitamisega seotud kirjed ja käskudega seotud kirjed [6]. Järgnevas peatükis on Thonny logifailide struktuuri kirja panemisel lähtutud eespool mainitud töödest. Käesoleva bakalaureusetöö raames on täpsemalt uuritud automaatse analüüsi jaoks olulisi kirjeid.

Logifailide kirjete põhjal saab detailset informatsiooni Thonny kasutamise ja kasutaja programmeerimise protsessi kohta. Kui kasutaja näiteks sisestab või eemaldab teksti, avab uusi faile, minimeerib Thonny akna, vajutab aknas nuppe, käivitab või silub programmi, siis selle kõige kohta on informatsioon ajatemplitega olemas. Samuti on võimalik selgeks teha, kas tegevus toimus programmiaknas või käsuaknas. Põhjalike andmete abil on võimalik tagantjärgi logifaili vaadates aru saada, mis programmis toimus, ning seda analüüsida.

Analüüsi tegemiseks on oluline mõista logifailide struktuuri ning osata tõlgendada kirjetes leiduvat informatsiooni. Logifailide uurimise tulemusena tutvustatakse järgnevates peatükkides kirjeid, mida kasutatakse automaatse analüüsi koostamisel. Kirjed koosnevad võti-väärtus (ingl *key-value*) paaridest ning nende hulk võib olenevalt kirje tüübist varieeruda. Kirjete seletuste juures tuuakse välja need võti-väärtus paarid, mis on olulised analüüsi jaoks, ning muid paare lahti ei seletata.

Järgnevates peatükkides selgitatakse põhjalikumalt analüüsis kasutatud kirjeid, mis on olulised järgmiste sündmuste tuvastamiseks: teksti sisestamine ja kustutamine, veateated ning programmi käivitamised.

3.2.1 Teksti sisestamine ja kustutamine

Logifailides on palju kirjeid teksti sisestamise ja kustutamise ning kleepimise ja kopeerimise kohta, kuna seda teeb Thonny kasutaja tavaliselt kõige rohkem. Need tegevused kuuluvad programmikoodiga seotud kirjete alla. Kui kasutaja kirjutab programmi koodi, siis iga sisestatud teksti kohta salvestatakse logifaili kirje, mis koosneb seitsmest võti-väärtus paarist:

1) *"index": <reanumber . mitmes element reas>* näitab, mis positsioonile sümbol(id) sisestati. Arv enne punkti tähistab rida, millele sümbol(id) lisati, kusjuures lugemine hakkab ühest. Arv pärast punkti näitab, mitmes on sisestatud sümbol(id) reas, kusjuures lugemine hakkab nullist.
2) *"text": <sümbol(id)>* näitab, mis sümbol(id) sisestati.
3) *"tags": <sildid>* näitab, mis siltidega on seda kirjet iseloomustatud.

4) *"text_widget_class": <element>* näitab, millisesse Thonny akna elementi on tekst sisestatud.
5) *"text_widget_id": <id>* näitab, millisesse Thonny akna kindlasse elementi sümbol(id) sisestati. Igal elemendil on oma id, kuna ühes Thonny aknas võib olla avatud näiteks mitu tekstiredaktorit (CodeViewText).
6) *"sequence": "TextInsert"* näitab, et tegemist on teksti sisestamisega.
7) *"time": <aeg>* näitab, mis ajahetkel tegevus aset leidis.

Kui programmeerija kirjutab programmikoodi tähthaaval, siis tuleb iga tähe kohta üks kirje (joonis 4). Teksti sisestamine võib toimuda nii tekstiredaktoris (CodeViewText) kui ka käsuaknas (ShellText). Kui teksti sisestatakse käsuaknas, siis esineb kirjes veel kaheksas võti-väärtus paar *"text_widget_context": "shell"*. Kui on soov uurida kasutaja sisestatud tähemärkide arvu, siis tuleks vaadelda just tekstiredaktoris sisestatud tähemärkide arvu, kuna käsuaknas ilmuvad nii Thonny genereeritud tekstid kui ka kasutaja sisestused programmi töö ajal.


```
{
    "index": "3.9",
    "text": "j",
    "tags": "None",
    "text_widget_id": 57822704,
    "text_widget_class": "CodeViewText",
    "sequence": "TextInsert",
    "time": "2019-03-24T12:30:56.652261"
},
```


Joonis 4. Ühe sümboli lisamine.

Kui Thonny kasutaja kleebib teksti, siis ilmub logifaili kirje, kus on näha tervet kasutaja sisestatud teksti ühes kirjes (joonis 5). Samasugune kirje ilmub ka siis, kui Thonnys avatakse juba varem loodud programmifail. Nimelt täidab Thonny ühe sisestusega Pythoni programmifaili koodiga tekstiredaktori, kus varem loodud programmifail avatakse. Selleks, et

olla kindel, et tegemist on kasutaja kleebitud tekstiga, tuleb kontrollida, kas kirjele järgneb kirje võti-väärtus paariga *"sequence": "<<Paste>>"* (joonis 6).


```
{
    "index": "2.0",
    "text": "print(\"tere\")",
    "tags": "None",
    "text_widget_id": 57822704,
    "text_widget_class": "CodeViewText",
    "sequence": "TextInsert",
    "time": "2019-03-24T12:30:20.097937"
},
```


Joonis 5. Mitme elemendi lisamine korraga.


```
{
    "widget_id": 70886640,
    "widget_class": "CodeViewText",
    "text_widget_id": 70886640,
    "text_widget_class": "CodeViewText",
    "sequence": "<<Paste>>",
    "time": "2017-12-04T17:15:05.977969"
},
```


Joonis 6. Kleepimise kirje.

Kui kasutaja kopeerib teksti mõnes tekstiredaktoris, siis selle kohta tekib logifaili kirje. See on väga sarnane kleepimise kirjele (joonis 6), kuid võti-väärtus paari *"sequence": "<<Paste>>"* asemel on *"sequence": "<<Copy>>"*. Kopeerimise puhul ei ole võimalik kindlaks teha, mis tekst kopeeriti.

Kui kasutaja kustutab teksti, siis iga järjest kustutatud teksti kohta salvestatakse logifaili kirje (joonis 7). Selleks, et olla kindel, et tegemist on kustutamisega, peab kirje sisaldama võti-väärtus paari *"sequence": "TextDelete"*.


```
{
    "index1": "6.0",
    "index2": "6.4",
    "text_widget_id": 70886640,
    "text_widget_class": "CodeViewText",
    "sequence": "TextDelete",
    "time": "2017-12-04T17:15:09.051174"
},
```


Joonis 7. Kustutamise kirje.

Kustutamise kirje koosneb kuuest võti-väärtus paarist:

1) *"index1": <reanumber . mitmes element reas>* näitab, mis positsioonist alates sümbol(id) kustutati.
2) *"index2": <reanumber . mitmes element reas>* näitab, mis positsioonini sümbol(id) kustutati (viimast elementi ei kaasata). Kui kustutatakse ainult üks sümbol, siis väärtus on *"None"*.
3) *"text_widget_class": <element>* näitab, millises Thonny akna elemendis on tekst kustutatud.
4) *"text_widget_id": <id>* näitab, millises Thonny akna kindlas elemendis sümbol(id) kustutati. Igal elemendil on oma id, kuna ühes Thonny aknas võib olla avatud näiteks mitu tekstiredaktorit (CodeViewText).
5) *"sequence": "TextDelete"* näitab, et tegemist on teksti kustutamisega.
6) *"time": <aeg>* näitab, mis ajahetkel tegevus aset leidis.

Siinkohal on samuti oluline, et vaadeldakse just tekstiredaktoris tehtud kustutamisi. Kuna tegevus võib toimuda ka käsuaknas, siis tuleb kontrollida, ega kirjes ei esine veel kaheksas võti-väärtus paar *"text_widget_context": "shell"*.

3.2.2 Veateated

Kui on vajadus uurida Thonny kasutajatel esinenud probleeme programmeerimise ajal, siis tuleks uurida veateateid, mis võivad tulla programmi käivitamisel, silumisel või käsureal tehtud tegevuste järel. Iga sisestatud veateate kohta salvestatakse logifaili kirje, mis koosneb kaheksast võti-väärtus paarist (joonis 8).

```
{
    "index": "975.0",
    "text": "SyntaxError: invalid syntax\n",
    "tags": "('io', 'stderr')",
    "text_widget_id": 88494992,
    "text_widget_class": "ShellText",
    "sequence": "TextInsert",
    "text_widget_context": "shell",
    "time": "2019-03-21T00:56:27.187825"
},
```

Joonis 8. Veateate kuvamine käsuaknas.

Veateate kirjes on analüüsiks olulised järgnevad võti-väärtus paarid:

1) *"text": <veateade>* näitab, mis veateade kasutajale kuvati.
2) *"tags": <sildid>* näitab, mis siltidega on seda kirjet iseloomustatud.
3) *"sequence": "TextInsert"* näitab, et tegemist on teksti sisestamisega.
4) *"text_widget_context": "shell"* näitab, et tegemist on käsuaknas tehtud tegevusega.
5) *"time": <aeg>* näitab, mis ajahetkel tegevus aset leidis.

Tegelikkuses on tegemist käsuaknas teksti sisestamise kirjega (vt ptk 3.2.1), millel on kindlad omadused. Esiteks on võtme *"text"* väärtuseks vea kohta kuvatud informatsioon, kus on olemas nii veatüüp kui ka -teade. Teiseks peab võtme *"tags"* väärtuste hulgas olema silt *'stderr'* (süntaksiviga) või *'error'* (muud vead).

3.2.3 Käivitamistega seotud kirjed

Thonny logifailide põhjal on võimalik teada saada, kui mitu korda ja mis ajahetkedel programm käivitati. Käivitamisel luuakse kirje (joonis 9), kus on informatsioon, mis programm ja mis ajahetkel käivitati.

```
{
    "command_text": "%Run proov.py\n",
    "sequence": "ShellCommand",
    "time": "2019-03-26T12:13:13.171595"
},
```

Joonis 9. Programmi käivitamise kohta tekkiv kirje.

Käivitamise kirje koosneb kolmest võti-väärtus paarist:

1) *"command_text": "%Run <failinimi>\n"* näitab, mis programmifail käivitati.
2) *"sequence": "ShellCommand"* näitab, et tegemist on käsuakna käsuga.
3) *"time": <aeg>* näitab, mis ajahetkel tegevus aset leidis.

Samuti saab koguda lisainformatsiooni, mis on seotud just käivitamistega. Üheks võimaluseks on kontrollida, kas programmi töö oli edukas ning kas esines programmikoodis vigu enne programmi töö lõppu või juba käivitamisel. Kui programm lõpetab töö, siis ilmuvad Thonny käsuaknasse sümbolid *">>> "* ja selle kohta logifaili kirje (joonis 10).

```
{
    "index": "2.0",
    "text": ">>> ",
    "tags": "('toplevel', 'prompt')",
    "text_widget_id": 88494992,
    "text_widget_class": "ShellText",
    "sequence": "TextInsert",
    "text_widget_context": "shell",
    "time": "2019-03-21T00:31:56.391809"
},
```


Joonis 10. Tekst, mille Thonny sisestab käsuaknasse peale programmi töö lõppu.

Käivitamisele järgnes viga, kui peale käivitamise kirjet (joonis 9) ja enne sümbolite ″>>> ″ kirjet esines mõni veateate kirje (ptk 3.2.2). Käivitamisele järgnevaid veateateid analüüsides tuleb arvestada alates Thonny versioonist 3.0 lisatud aknaga “Assistant”, mille avanemise kohta võib tekkida kirje sümbolite ″>>> ″ kirje ja veateate kirje vahele erinevalt varasematest versioonidest, kus veateate korral veateate kirje esines täpselt enne sümbolite ″>>> ″ kirjet.

4. Thonny logifailide analüsaator

Järgnevas peatükis antakse ülevaade loodud tarkvarast. Täpsemalt käsitletakse tarkvara kasutamist, funktsionaalsuseid ja kogutud tagasisidet. Tagasiside tarkvarale on saadud Tartu Ülikooli aine "Programmeerimise alused" õppejõududelt.

Tarkvara loomiseks kasutati siinses töös Pythonit, kuna see keel tagab suurima tõenäosusega loodud programmi edasise arendamise. Nimelt on Python keel, mida valdavad kõik rakenduse kasutajad, kuna sihtgrupiks on Pythoni õpetajad ja õppejõud ning suure tõenäosusega oskavad seda ka teised, kellel on plaan Thonny logifailide analüüsimist veelgi edasi arendada. Seega saavad tulevased kasutajad lisada uusi funktsionaalsusi valminud lahendusele.

Kasutajaliidese loomiseks kasutati Tkinterit. Tegemist on kergekaalulise Pythoni standardvarustusse kuuluva mooduliga, millega saab suhteliselt mugavalt luua lihtsat kasutajaliidest [9]. Samuti toimib see moodul mitmetel operatsioonisüsteemidel nagu Linux, Windows ja Mac OS [10]. Kasutajaliideses graafikute näitamiseks kasutati teeki Matplotlib, millega saab graafikud integreerida Tkinteriga loodud kasutajaliidesesse.

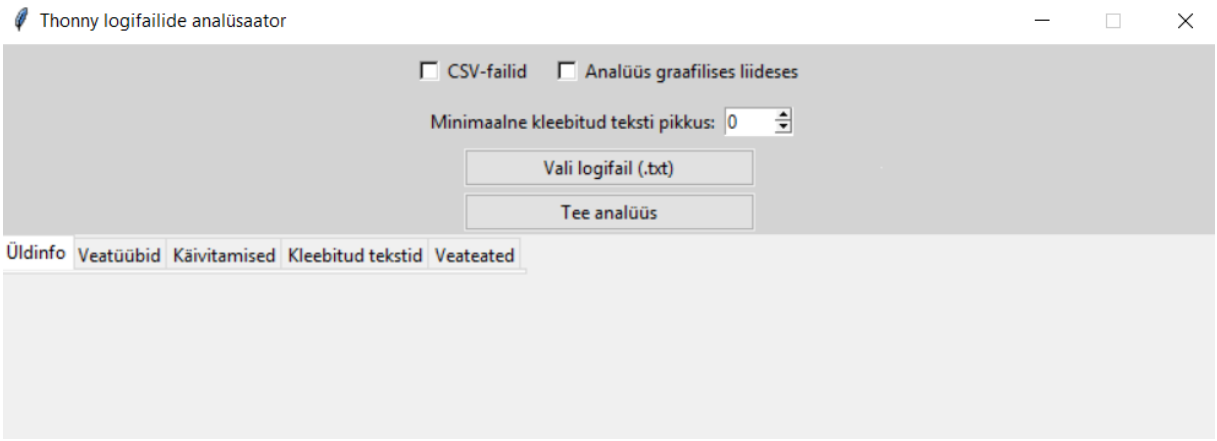
4.1 Tarkvara funktsionaalsused ja kasutamine

Andmete kogumiseks vaadatakse läbi kogu logifail ja otsitakse analüüsi jaoks olulisi kirjeid (ptk 3.2). Analüüsi käigus koostatakse andmetest sõnastik, mille põhjal on võimalik vastavalt kasutaja valitud väljundile kuvada graafikud või kirjutada analüüsi käigus kogutud andmed faili.

Tarkvaral on kaks väljundit: kasutajaliideses kuvatav üldine informatsioon ja graafikud ning CSV-failid (ingl *Comma-separated values*), kuhu salvestatakse analüüsi põhjal kogutud andmed. Esimene väljund on oluline selleks, et oleks võimalik kiiresti saada informatsiooni analüüsitud logifaili kohta. Teine väljund on mõeldud selleks, et analüüsi põhjal kogutud andmeid salvestada ning siis on võimalik CSV-faile täpsemalt uurida või teha edasist töötlust andmetega. Tarkvara on püütud koostada pidades silmas lihtsust ja kasutusmugavust.

Programmi käivitamisel avaneb kasutajaliides (joonis 11), kus on võimalik valida väljund. Väljundiks on kas üldine informatsioon ja graafikud kasutajaliideses või CSV-failide genereerimine. Esimese väljundi valimiseks tuleb märkida „Analüüs graafilises liideses" ja

teise puhul „CSV-failid“. Võimalik on valida mõlemad väljundid korraga, kuid vähemalt üks peab olema analüüsi tegemiseks valitud.



Joonis 11. Thonny logifailide analüüsi programmi kasutajaliides.

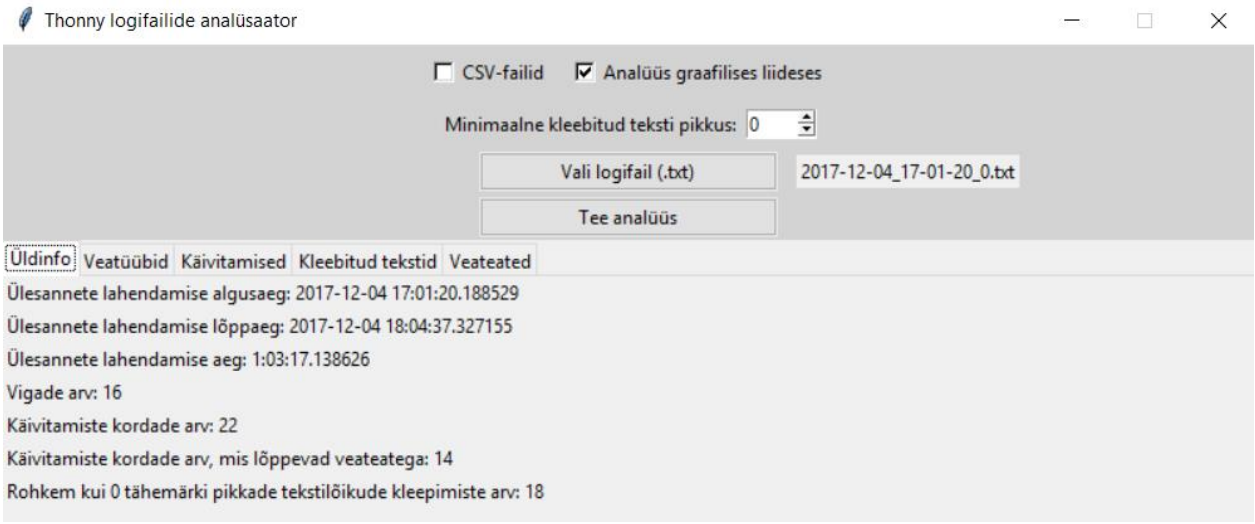
Lisavõimalusena saab määrata minimaalse kleebitud teksti pikkuse, millest lühemad kleebitud tekstid eemaldatakse ja neid analüüsis ei vaadelda. Kasutajal on seega võimalus välja jätta näiteks tekstid, mis on lühemad kui 20 tähemärki, kui on soov vaadelda ainult pikemaid kleebitud tekste. Vaikeväärtusena on minimaalse kleebitud teksti pikkuseks null ehk vaadeldakse kõiki kleebitud tekste. Klõpsates nupul „Vali logifail (.txt)“ avaneb failipuu ja kasutajal on võimalik valida txt-fail. Klõpsates nupul “Tee analüüs” tehakse analüüs, kui varem oli valitud vähemalt üks analüüsi väljund ja tegemist on korrektse logifailiga. Kui kasutajal on soov teha analüüsi mõne teise faili põhjal, siis tuleb uuesti klõpsata nupule „Vali logifail (.txt)“ ning on võimalik valida uus logifail.

4.1.1 Analüüs graafilises liideses

Kui valitud väljundiks on analüüs graafilises liideses, siis kuvatakse kasutajaliideses eri alamlehtedel analüüsi käigus kogutud andmeid.

4.1.1.1 Alamleht „Üldinfo“

Alamlehel „Üldinfo“ (joonis 12) kuvatakse informatsioon, mis ei ole detailne, kuid võimaldab analüüsi uurijal juba teatud määral aru saada lahendamise protsessist. Selle informatsiooni põhjal on kasutajal võimalik otsustada, milliseid alamlehti või CSV-faile võiks olla kasulik uurida.



Joonis 12. Alamleht „Üldinfo“.

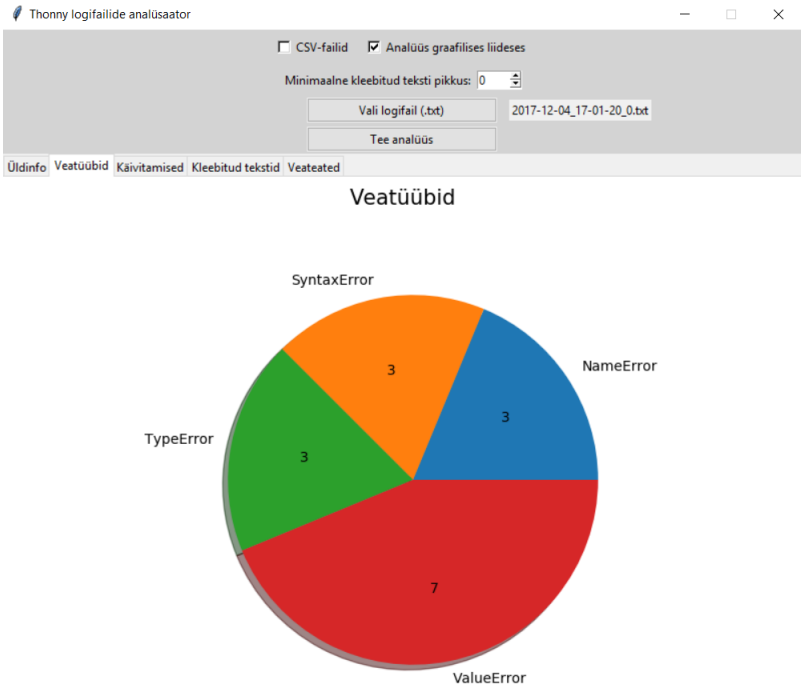
Esimesel alamlehel kuvatakse järgnev informatsioon:

1) Ülesannete lahendamise algusaeg – näitab, mis ajahetkel on alustatud ülesande lahendamist.
2) Ülesannete lahendamise lõppaeg – näitab, mis ajahetkel on lõpetatud lahendamine.
3) Ülesannete lahendamise aeg – näitab, kui palju aega kulus ülesannete lahendamiseks.
4) Vigade arv – näitab, mitu korda esines kontrolltöö lahendamisel viga.
5) Käivitamiste kordade arv – näitab, mitu korda on kontrolltöö jooksul faile käivitatud.
6) Käivitamiste kordade arv, mis lõppevad veateatega – näitab, mitu korda on kontrolltöö jooksul failide käivitamise tagajärjel esinenud veateade.
7) Rohkem kui n tähemärki pikkade tekstilõikude kleepimiste arv – näitab, mitu korda on kleebitud teksti, mis on vähemalt n tähemärki pikk.

Alamlehel esitatakse üldisem kokkuvõte andmetest, mille abil koostatakse järgnevatel alamlehtedel juba täpsemaid graafikuid ja tabeleid. Lisaks esitatakse lahendamise algus- ja lõppaeg ning ülesannete lahendamise aeg, mida järgnevatel alamlehtedel eraldi välja ei tooda.

4.1.1.2 Alamleht „Veatüübid“

Alamlehel „Veatüübid“ (joonis 13) kuvatakse sektordiagramm, kus on näha eri veatüüpide esinemise osakaal. Sektordiagrammi põhjal on võimalik selgelt näha, mis tüüpi veateateid on lahendajal enim esinenud. Enamasti esineb algajatel programmi loomisel kõige rohkem just süntaksi vigu. Kui veateateid ei esinenud, siis graafikut ei kuvata.

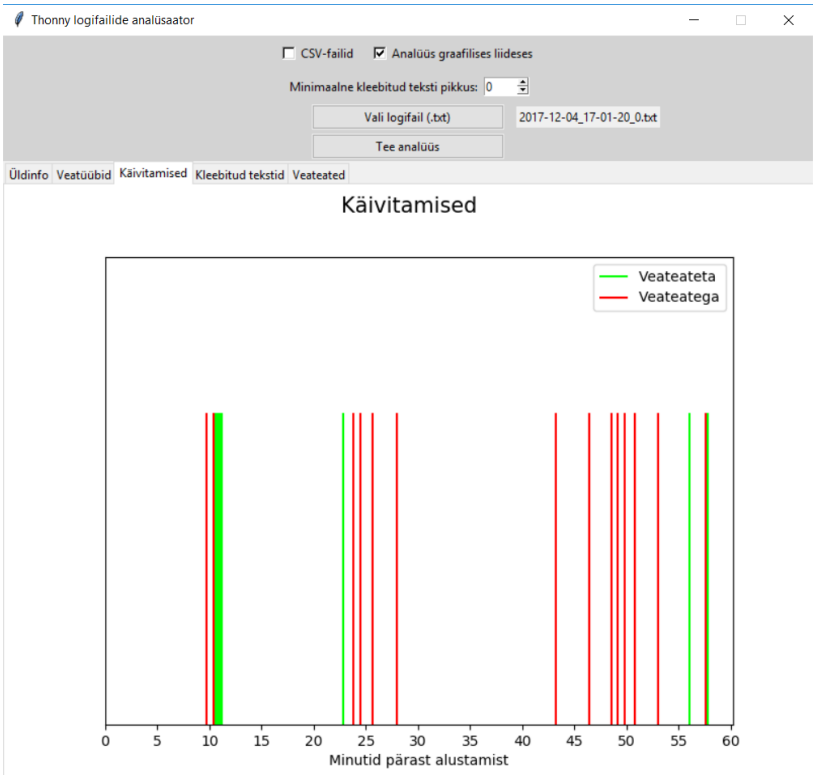


Joonis 13. Alamleht "Veatüübid".

Veatüüpide puhul tuuakse välja kõik veateated, mis esinesid programmi loomise käigus. Veateated võisid tekkida nii programmi töö, silumise või käsureal tehtud tegevuste käigus, mis ei pruugi olla seotud programmi tööga.

4.1.1.3 Alamleht „Käivitamised“

Alamlehel „Käivitamised“ (joonis 14) kuvatakse graafik, kus rohelise joonega on märgitud edukad ehk ilma veateateta käivitamised ja punasega on märgitud käivitamised, millele järgnes veateade. Kõik käivitamist märkivad jooned asuvad ajajoonel, tänu millele on võimalik näha, mitu minutit pärast alustamist on käivitamine toimunud. Graafiku abil on võimalik vaadelda, kui palju ja millisel ajahetkel on kasutaja enda loodud programmi seda käivitades testinud. Samuti saab näha, kas kasutajal õnnestus viga ära parandada või esines veateateid järjest.

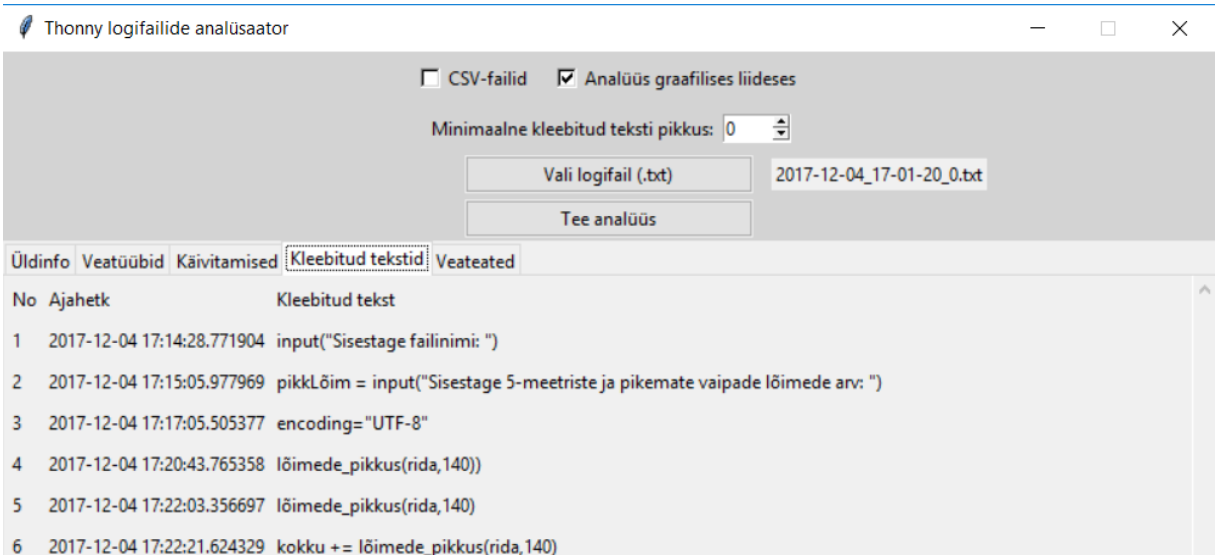


Joonis 14. Alamleht "Käivitamised".

Selle graafiku juures peab meeles pidama, et see esitab just käivitamisele järgnenud veateateid, kuid veateateid võib tulla ka näiteks silumise käigus. Kui kasutaja sulges Thonny programmeerimiskeskkonna programmi töö ajal, siis viimase käivituse kohta ei ole täielikku informatsiooni, kas käivitus lõppes veateatega või mitte. Seetõttu sellises olukorras jäetakse käivitus analüüsist välja. Kasutajal on võimalik käivitamiste kohta kogutud andmed saada CSV-failina "*failinimi*_runnings.csv".

4.1.1.4 Alamlehed tabelitega

Lisaks üldisele informatsioonile ja graafikutele on veel alamlehed „Kleebitud tekstid" (joonis 15) ja „Veateated", kus kuvatakse sama informatsioon, mis CSV-failides "*failinimi*_pasting.csv" ja "*failinimi*_errors.csv" (vt ptk 4.1.2). Need on mõeldud selleks, et kasutaja ei peaks CSV-faile alla laadima, kui ei ole soovi nendega midagi hiljem edasi teha, ja saaks näha täpsemat informatsiooni veateadete ja kleepimiste kohta ka kasutajaliideses.



Joonis 15. Alamleht "Kleebitud tekstid".

Kleebitud tekstide hulka arvestatakse ainult tekstid, mis kleebiti tekstiredaktoris, ning veateadete hulka arvestatakse kõik programmi loomise käigus tekkinud veateated.

4.1.2 CSV-failid

CSV-failid on mõeldud selleks, kui on soov analüüsi käigus kogutud andmed salvestada ning teha nendega edasist töötlust. Kui valitud väljundiks on CSV-failid, siis genereeritakse need samasse kausta, kus asub vaadeldav logifail "*failinimi*.txt". Andmete põhjal luuakse kolm faili: "*failinimi*_errors.csv", "*failinimi*_pasting.csv" ja "*failinimi*_runnings.csv".

Esimeses failis ("*failinimi*_errors.csv") on andmed vigade kohta ning faili esimeses reas ehk pealkirjade reas on väärtused *error_time*, *error_type* ja *error_message* (joonis 16). Esimeses veerus on vigade esinemise aeg ajatemplina, teises veerus on veatüüp ja kolmandas veerus täpne veateade.

	A	B	C
1	error_time	error_type	error_message
2	2017-12-04 17:10:02.744640	NameError	name 'lõimede_pikkus' is not defined
3	2017-12-04 17:11:05.363150	SyntaxError	unexpected EOF while parsing
4	2017-12-04 17:11:43.786017	SyntaxError	unexpected EOF while parsing
5	2017-12-04 17:25:31.169461	NameError	name 'failinimi' is not defined
6	2017-12-04 17:26:05.676721	TypeError	'<' not supported between instances of 'str' and 'int'
7	2017-12-04 17:27:12.585238	ValueError	could not convert string to float: '\ufeff7\n'
8	2017-12-04 17:29:29.569077	ValueError	could not convert string to float: '\ufeff7'
9	2017-12-04 17:34:41.871021	ValueError	could not convert string to float: '\ufeff7'
10	2017-12-04 17:44:49.352681	ValueError	could not convert string to float: '\ufeff4,9'
11	2017-12-04 17:47:55.429805	ValueError	could not convert string to float: '\ufeff4,9'
12	2017-12-04 17:50:01.634025	NameError	name 'puhasRida' is not defined
13	2017-12-04 17:50:41.460894	ValueError	could not convert string to float:
14	2017-12-04 17:51:19.852561	ValueError	could not convert string to float: '4,9'
15	2017-12-04 17:52:15.170258	TypeError	can't multiply sequence by non-int of type 'float'
16	2017-12-04 17:54:30.594094	TypeError	can't multiply sequence by non-int of type 'float'
17			

Joonis 16. CSV-fail veateadete kohta.

Teises failis (“*failinimi*_pasting.csv”) on andmed kleebitud tekstide kohta ning faili pealkirjade reas on väärtused *pasted_text_time* ja *pasted_text*. Esimeses veerus on teksti kleepimise ajahetk ajatemplina ja teises veerus on kleebitud tekst.

Kolmandas failis (“*failinimi*_runnings.csv”) on andmed programmi käivitamiste kohta ning pealkirjade reas on väärtused *running* ja *running_results_in_error*. Esimeses veerus on käivitamise ajahetk ajatemplina ning teises veerus on tõeväärtus, mis näitab, kas käivitamisele järgnes veateade või ei.

4.2 Tagasiside tarkvarale

Arendamise juures oli oluline saadav tagasiside, mille põhjal sai teha parandusi ja edasiarendusi. Töö käigus andsid juhendajad tagasisidet tarkvarale, mille tulemusena sai loodud tarkvara juba parandada. Kui esimene versioon sai valmis, siis andsid tagasisidet kursuse “Programmeerimise alused” kolm õppejõudu, kes testisid programmi kontrolltöös kogutud logifailide põhjal. Testijad said GitHubi lingi (Lisa I) ning küsimustiku, mida anonüümselt täita pärast logifailide analüüsimist rakendusega. Küsimustiku eesmärgiks oli leida vigu ja probleeme tarkvara töös, arendada seda edasi ning mõista, kas praegu loodud tarkvara täidab oma eesmärki õppejõu abistamisel.

Küsimustikus oli kahte liiki küsimusi – osale sai vastata vabas vormis ning teistele 5-palli skaalal. Selleks, et mõista, kas rakendus on õppejõule kasulik ja mugav kasutada ning millised funktsionaalsused on olulised, olid küsimustikus järgmised küsimused:

1) Kui lihtne oli programmi kasutada? (1 on “väga raske” ja 5 on “väga lihtne”)
2) Milline on teie üldine hinnang rakendusele? (1 on “väga halb” ja 5 on “väga hea”)
3) Milliseid rakenduse võimalusi peate ebaoluliseks?
4) Milliseid rakendusi võimalusi peate kasulikuks?
5) Kui palju kasutaksite rakendust ka tulevikus?

Esimesele küsimusele valis üks inimene vastuseks 3 ja kaks inimest 5. Teisele küsimusele valisid kõik kolm vastatajat vastuseks 4. Pärast neid küsimusi tuli veel mitmeid küsimusi, kus nad said kahes eelmainitud küsimuses vastuse valikut põhjendada vastates sisulistematele küsimustele ning soovitades parandusi. Kolmanda küsimuse juures ei osanud kaks vastajat ühtegi ebaolulist aspekti välja tuua. Üks vastaja pidas aga ebaoluliseks veatüüpide sektordiagrammi, mis kontrolltöö logifailide analüüsimise juures pigem tõesti palju juurde ei anna. Samuti pidas ta ebaoluliseks üldinfo alamlehel olevat käivitamiste kordade arvu, mis lõppevad veateatega, kuna see on enamasti null. See tulenes rakenduses olevast veast, mis nüüdseks on parandatud. Neljanda küsimuse vastused erinesid juba rohkem, kuid nende põhjal on võimalik välja tuua enam kasulikuks peetud võimalusi: kleebitud tekstid ning CSV-faili salvestamise võimalus. Üldiselt tõid vastajad selle küsimuse juures üle poole funktsionaalsustest välja ning ei olnud ühtegi funktsionaalsust, mida ei oleks kordagi mainitud. Viienda küsimuse juures arvasid kõik vastajad, et kasutaksid rakendust ka tulevikus. Üks vastaja täpsustas, et kasutaks programmi tööst esialgse mulje saamiseks, ja teine kirjutas, et kasutaks programmi nii üldpildi saamiseks kui ka plagiaadi kontrollimiseks.

Mõistmaks, mis funktsionaalsusi oleks veel võimalik lisada, mida võiks teha teisiti ning kas rakenduses esineb vigu, esitasin küsimustikus järgmised küsimused:

1) Milliseid võimalusi sooviksite rakenduses veel näha?
2) Mis võiks rakenduses olla teisiti?
3) Kas leidsite programmi töös vigu? Kui jah, siis milliseid?
4) Palun kirjutage veel kommentaare.

Esimese ja teise küsimuse puhul toodi välja palju mõtteid, millest osa sai pärast tagasisidega tutvumist rakendusse lisatud ja ülejäänud toon täpsemalt välja peatükis “Edasised võimalused”. Pärast tagasisidega tutvumist parandati programmi vastavalt järgmistele soovitustele:

1) Kui logifaili lugemisel tekivad vead, siis kuvatakse see liideses.
2) Logifaili avamisel jäetakse meelde jooksev kataloog.
3) Sama logifaili põhjal on võimalik teha mitu analüüsi järjest, muutes ainult väljundit.
4) Peale CSV-faili salvestamist antakse kasutajale märku, et failid on salvestatud ja kuhu.
5) Rakenduses on kohe aktiivne valik "Analüüs graafilises liideses".

Soovitused, mida on võimalik hilisemate arenduste käigus rakendusele lisada:

1) Tuua välja käsurea väljund (täpsemalt print käsud).
2) Kuvada kasutajaliideses lõplik programm ja võimalusel vahekujud. Täpsemalt kuvada koodi muutumise dünaamikat.

Kolmanda küsimuse juures, kus palusin tuua välja vigu programmi töös, tõid kaks testijat välja sama probleemi. Nimelt analüüsi järgi ei esinenud enamasti kasutajal veateateid pärast käivitamist, kuigi üldinfo alamlehel oli näha, et kasutajal oli esinenud palju veateateid. Viga tuli sellest, et alates Thonny versioonist 3.0 lisatud aken "Assistant" võib tekitada akna avanemise kohta kirje veateate kirje ja programmitöö lõpu kirje vahele, mistõttu need kirjed ei pruugi enam järjest esineda (ptk 3.2.3). Lõpukommentaarina tõi üks vastaja välja, et "programmiga ei saa vahet teha headel ja kehvadel töödel, sest vigaseid käivitusi ei tooda välja". See tähendab, et on olemas veel üks võimalus edasi arenduseks – tuua välja iga vigase käivituse kohta täpsemat informatsiooni.

4.3 Edasised võimalused

Loodud töövahendi abil on võimalik teha logifailide põhjal analüüs, kuid on veel mitmeid võimalusi koostatud analüüsi parandamiseks. Saadud tagasiside põhjal saab järeldada, et oluline oleks tuua välja rohkem programmi koodi. Täpsemalt nii programmi lõppkuju kui ka vahekujud koos veateadete ja kleebitud tekstidega. See aitaks veel paremini mõista õppijal ülesande lahendamise käigus tekkinud probleeme. Programmi vahekujude jälgimiseks tuleks vaadelda teksti lisamise ja kustutamise kirjeid, kuid tuleb jälgida, millises tekstiredaktoris muudatused tehti. Nimelt võib kasutajal olla korraga Thonny programmeerimiskeskkonnas lahti rohkem kui üks tekstiredaktor. Lisaks võiks välja tuua tähemärkide hulga muutuste graafiku, millelt oleks võimalik näha näiteks järske muutusi tähemärkide hulgas.

Hetkel vaadeldakse mitme ülesande lahendamise protsessi ühena, kui nende lahenduskäigud on kõik ühes logifailis. Selline lähenemine on küll hea näiteks kontrolltöö analüüsimiseks, kuid

vahel on parem, ka kontrolltöö puhul, vaadelda informatsiooni ainult ühe ülesande kohta. Selleks oleks vaja lisada funktsionaalsus, mille abil saaks automaatselt eraldada logifaili ülesannete põhjal mitmeks logifailiks ning siis nende põhjal teha analüüs.

5. Kokkuvõte

Käesoleva bakalaureusetöö eesmärk oli luua tarkvaralahendus, mis suudaks automaatselt analüüsida Thonny logifaile. Genereeritud analüüsi põhjal on uurijal või õppejõul võimalik teha järeldusi õppijate käitumismustrite, õppemetoodika ja õppematerjalide kohta.

Töös kirjeldati programmeerimiskeskkonda Thonny ja Thonny logifaile, täpsemalt just automaatse analüüsi jaoks olulisi kirjeid. Samuti tutvustati bakalaureusetöö raames loodud tarkvaralahendust, mille abil on võimalik automaatselt analüüs koostada. Tarkvaral on kaks väljundit – graafiline liides, kus kuvatakse graafikute ja tabelite abil nii üldine, veateadete, kleebitud tekstide ja käivitamiste informatsioon, ning CSV-failid, kuhu salvestatakse analüüsi käigus kogutud andmed.

Tarkvara on testinud kursuse “Programmeerimise alused” õppejõud, kelle tagasiside põhjal saab väita, et tarkvara täidab eesmärki. Tarkvara on võimalik soovitatud edasiste võimaluste (ptk 4.4) põhjal edasi arendada, et analüüs oleks veelgi põhjalikum.

Viidatud kirjandus

[1] Tartu Ülikooli ÕIS. http://ois.ut.ee (02.01.2019)

[2] IT-kursused 2018. – 2019. aastal. Tartu Ülikooli arvutiteaduse instituut. http://programmeerimine.ut.ee/ (02.01.2019)

[3] Tartu Ülikooli statistika. https://www.ut.ee/et/sisseastumine/statistika-0 (02.01.2019)

[4] Guo, P. Python Is Now the Most Popular Introductory Teaching Language at Top U.S. Universities. Communications of the ACM, 2014. https://cacm.acm.org/blogs/blog-cacm/176450-python-is-now-the-most-popular-introductory-teaching-language-at-top-u-s-universities/fulltext (02.01.2019)

[5] Thonny. https://courses.cs.ut.ee/2016/eprogalused/Main/Thonny (02.01.2019)

[6] Pedel, K. (2016) E-kursuse "Programmeerimise alused" logifailide analüüs (Bakalaureusetöö). Tartu Ülikool, arvutiteaduse instituut. http://comserv.cs.ut.ee/ati_thesis/datasheet.php?id=53665&year=2016 (27.01.2019)

[7] Meier, H. (2018) Õppijate käitumismustrid programmeerimisülesande lahendamisel: logifailide analüüs (Magistritöö). Tartu Ülikool, arvutiteaduse instituut. https://comserv.cs.ut.ee/ati_thesis/datasheet.php?id=62175&year=2018 (02.04.2019)

[8] Introducing JSON. http://www.json.org/ (30.04.2019)

[9] Tkinter. Programmeerimise õpik. https://progeopik.cs.ut.ee/tkinter.html (19.03.2019)

[10] TkDocs. https://tkdocs.com/ (19.03.2019)

Lisad

I. Thonny logifailide analüsaator

Thonny logifailide automaatse analüüsi programmikood ja logifail programmi töö katsetamiseks on kättesaadav aadressilt https://github.com/ageroosi/thonny-logfile-analysation.

II. Litsents

Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks

Mina, **Age Roosi**,

1. annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) minu loodud teose

   Thonny logifailide analüüsi automatiseerimine,

mille juhendajad on Eno Tõnisson ja Heidi Meier,

reprodutseerimiseks eesmärgiga seda säilitada, sealhulgas lisada digitaalarhiivi DSpace kuni autoriõiguse kehtivuse lõppemiseni.

2. Annan Tartu Ülikoolile loa teha punktis 1 nimetatud teos üldsusele kättesaadavaks Tartu Ülikooli veebikeskkonna, sealhulgas digitaalarhiivi DSpace kaudu Creative Commonsi litsentsiga CC BY NC ND 3.0, mis lubab autorile viidates teost reprodutseerida, levitada ja üldsusele suunata ning keelab luua tuletatud teost ja kasutada teost ärieesmärgil, kuni autoriõiguse kehtivuse lõppemiseni.

3. Olen teadlik, et punktides 1 ja 2 nimetatud õigused jäävad alles ka autorile.

4. Kinnitan, et lihtlitsentsi andmisega ei riku ma teiste isikute intellektuaalomandi ega isikuandmete kaitse õigusaktidest tulenevaid õigusi.

Age Roosi

10.05.2019